09/000924

WEST

Help | Logout | Interrupt

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers | Preferences

Search Results -

Terms	Documents
l11 not l10	6

Database: US Patents Full-Text Database; US Pre-Grant Publication Full-Text Database; EPO Abstracts Database; JPO Abstracts Database; Derwent World Patents Index; IBM Technical Disclosure Bulletins

Refine Search: l11 not l10

Clear

Search History

Today's Date: 1/16/2002

DB Name	Query	Hit Count	Set Name
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l11 not l10	6	L12
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	authori$ and (compoun$ adj (document$ or data)) and (link$ with embed$ with object$)	7	L11
PGPB,JPAB,EPAB,DWPI,TDBD	(authori$ and (compoun$ adj document$))	3	L10
PGPB,JPAB,EPAB,DWPI,TDBD	(authori$ and (compoun$ adj document$1))	3	L9
USPT	(authori$ and (compoun$ adj document$1)).clm.	0	L8
USPT	(authori$ and (compoun$ adj data)).clm.	0	L7
JPAB,EPAB,DWPI,TDBD	(authori$ and (compoun$ adj data)).clm.	0	L6
JPAB,EPAB,DWPI,TDBD	((ole or (object$ with link$ with embed$)) and (compoun$ adj data)).clm.	0	L5
JPAB,EPAB,DWPI,TDBD	(ole or (object$ with link$ with embed$)) and authori$ and (compoun$ adj data) and (input$ adj data)	0	L4
USPT	((ole or (object$ with link$ with embed$)) and authori$ and (compoun$ adj data)).clm.	0	L3
USPT	((ole or (object$ with link$ with embed$)) and authori$ and (input adj data)).clm.	0	L2
USPT	((ole or (object$ with link$ with embed$)) and authori$ and data and (input adj data)).clm.	0	L1

reviewed

considered

PALM INTRANET

Day : Wednesday
Date: 1/16/2002
Time: 14:35:47

Foreign Information for 09/000924

Priority#	Date	Country
9-154046	06/11/1997	JAPAN

Appln Info | Contents | Petition Info | Attorney/Agent Info | Continuity Data | Foreign Data | Inv

Search Another: Application# [ ] Search or Patent# [ ] Search

PCT / [ ] / [ ] Search or PG PUBS # [ ] Search

Attorney Docket # [ ] Search

(To Go BACK Use BACK Button on Your BROWSER Tool Bar)

Back to PALM | ASSIGNMENT | OASIS | Home page

WEST

Generate Collection

L12: Entry 1 of 6 File: USPT Aug 21, 2001

US-PAT-NO: 6279112
DOCUMENT-IDENTIFIER: US 6279112 B1

TITLE: Controlled transfer of information in computer networks

DATE-ISSUED: August 21, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
O'Toole, Jr.; James W.	Cambridge	MA		
Gifford; David K.	Weston	MA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Open Market, Inc.	Cambridge	MA			02

APPL-NO: 8/ 741862
DATE FILED: October 29, 1996

INT-CL: [7] G06F 11/30
US-CL-ISSUED: 713/201; 705/14
US-CL-CURRENT: 713/201; 705/14
FIELD-OF-SEARCH: 395/187.01, 395/188.01, 395/200.59, 380/21, 380/23, 380/24, 380/25, 713/154, 705/14, 705/51, 705/57, 705/59, 705/77, 705/80

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5341293	August 1994	Vertelney et al.	N/A
☐	5347632	September 1994	Filepp et al.	N/A
☐	5450593	September 1995	Howell et al.	N/A
☐	5455953	October 1995	Russell	395/739
☐	5490244	February 1996	Isensee et al.	N/A
☐	5586260	December 1996	Hu	395/200.2
☐	5594921	January 1997	Pettus	N/A
☐	5617565	April 1997	Augenbraun et al.	N/A
☐	5673322	September 1997	Pepe et al.	N/A
☐	5680452	October 1997	Shanton	N/A
☐	5710918	January 1998	Lagarde et al.	N/A
☐	5715314	February 1998	Payne et al.	380/24
☐	5717923	February 1998	Dedrick	395/613
☐	5724424	March 1998	Gifford	380/24
☐	5761648	June 1998	Golden et al.	705/14
☐	5809242	September 1998	Shaw et al.	395/200.47
☐	5838790	November 1998	McAuliffe et al.	380/4
☐	5948061	September 1999	Merriman et al.	709/219

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
WO 97/15885	May 1997	WOX	

OTHER PUBLICATIONS

Open Market, Inc.; OM Express.TM. Information Area; http://www.openmarket.com/express; Jul. 29, 1996.

ART-UNIT: 275
PRIMARY-EXAMINER: Beausoliel, Jr.; Robert W.
ASSISTANT-EXAMINER: Baderman; Scott T.
ATTY-AGENT-FIRM: Fish & Richardson P.C.

ABSTRACT:

The present invention relates to techniques for controlling transfers of information in computer networks. One technique involves transmitting from a server computer to a client computer a document containing a channel object corresponding to a communication service, and storing an access ticket that indicates that a user of the client computer permits the information source computer to communicate with the user over a specified channel. Another technique involves transmitting smart digital offers based on information such as coupons and purchasing histories stored at the computer receiving the offer. Another technique involves transmitting from a server computer to a client computer a request for a user's personal profile information, and activating a client avatar that compares the request for personal profile information with a security profile of the user limiting access to personal profile information. Another technique involves transmitting from a server computer to a client

computer a document containing an embedded link, activating the embedded link at the client computer and recording activation of the embedded link in a metering log.

62 Claims, 9 Drawing figures

L12: Entry 1 of 6 File: USPT Aug 21, 2001

DOCUMENT-IDENTIFIER: US 6279112 B1
TITLE: Controlled transfer of information in computer networks

BSPR:
U.S. patent application Ser. No. 08/168,519, filed Dec. 16, 1993 by David K. Gifford and entitled "Digital Active Advertising," the entire disclosure of which is hereby incorporated herein in its entirety by reference, describes a network sales or payment system that includes at least a client computer and a payment computer. The client computer transmits a payment order and an authenticator to the payment computer. The payment computer verifies the authenticator, transmits a payment authorization message and an authenticator back to the client computer, and performs a payment settlement transaction.

BSPR:
The client avatar acts as an agent for the user by controlling the release of information from the client personal profile to the server computer. The client avatar makes it possible to store a single client personal profile at the client computer or an agency computer, rather than multiple personal profiles at multiple server computers, while at the same time limiting the release of certain information from the personal profile only to trusted servers or only upon specific authorization from the user.

DEPR:
Referring to FIG. 1, a network-based system for controlled asynchronous transfer of information includes a client computer 10, operated by a user, that filters information transferred asynchronously to the client computer, a server computer 12 that transmits a document to the client computer containing a channel object that can be activated to authorize an asynchronous transfer of information, an information source computer 14 that asynchronously transfers the information, and an optional notification server 16 that acts as a trusted intermediary that filters asynchronously transferred information on behalf of the client computer. In certain implementations server computer 12 and information source computer 14 are the same computer. As used herein, the term "asynchronous" transfer of information refers to a transfer of information from an information source computer that is initiated by the information source computer rather than by another computer to which the information source computer responds.

DEPR:
The description of the asynchronous communication service in the channel object may include a certificate that includes an identification of the supplier of the information to be transmitted to the client computer, as well as the supplier's public key, the certificate being signed by a certifying authority. This public key will be used by the client computer to authenticate the information to be transmitted to the client computer by the information source computer.

DEPR:
When the document is displayed on the user computer, the icon contained in the channel object is displayed on the document as a representation of the channel object, and the user can determine from the document whether to authorize delivery of the content of the channel object as described in the document. The user can activate or select the channel object by clicking on a

representation of the channel object on the document, or a channel object in a document or broadcast received by the client computer may be activated automatically by the computer if the keywords or the other identifying information contained in the channel object match preset parameters pre-programmed into the client computer as a personal profile of the user (step 28). For example, the user may pre-program the computer to search for a keyword phrase such as "BUGS BUNNY" to automatically activate channel objects pertaining to BUGS BUNNY. Similarly, the user may authorize automatic activation of channel objects containing an embedded "G" rating, or automatic activation of only one megabyte of information per week.

DEPR:
Activation of the channel object causes an access ticket containing the description of the asynchronous communication service to be added to the client control list in the client computer, or causes the access ticket to be sent to the notification server, which adds it to the access control list (step 30). The access ticket permits the information source computer to communicate asynchronously with the client computer over a channel specified by the channel object, which may be a broadcast or multicast channel at a specific time period, or which may be the computer network linking the client computer and the information source computer in the event that the information from the information source computer is to be received by means of an asynchronous communication over the computer network. Thus, the activation of a channel object initiates an asynchronous communication channel from the information source computer to the client computer and instructs the client computer that the information source computer is authorized to send information over the channel.

DEPR:
Channel objects may be embedded not only in documents or pages on the World Wide Web, but in an alternative implementation they may be embedded in e-mail messages, OLE objects, ActiveX applets, etc. In fact, all of the communications between the server computer and the client computer and between the information source computer and the client computer may occur by e-mail, via compound documents, etc.

DEPR:
If the smart digital offer object attempts to observe the purchasing history or certain other user-specific information, the client computer asks the user whether the user wishes to reveal the information (step 122). The user indicates whether release of the information is authorized (step 124), and the smart digital offer object then examines the coupon (including the coupon's authenticator), digital receipts (including authenticators) and other user-specific information authorized to be revealed by the user, and presents to the user an offer of a product or service (step 126). The execution environment at the client computer can under some circumstances change between steps 118 and 126. For example, the client computer may receive a coupon after step 118 occurs but before step 126 occurs. In one particular embodiment the client computer includes a client "avatar" of the type described below in connection with FIGS. 5 and 6, which limits the release of certain information only to trusted servers, or only upon authorization from the client user, or both.

DEPR:
The terms or conditions of the offer, such as price and payment terms, are calculated by the smart digital offer object using formulas that depend on the information contained in the digital coupons and the other information examined by the smart digital offer object, including the time of day, or user profile information such as membership codes, user's age, user's income, and other demographic information certified by an independent authority with an authenticator. When the user accepts the offer (step 128) the client computer sends a message to the offer-providing server indicating that the user has accepted the offer, or sends the message to an intermediary server that is trusted by the client computer to maintain the confidentiality of user-specific information and is trusted by the offer-providing server to

verify the terms on which the offer was accepted (step 130). The message sent to the offer-providing server or the intermediary server includes the terms upon which the offer was accepted and also includes an authenticator. The offer-providing server or the intermediary server verifies the terms on which the offer was accepted by verifying the authenticator (step 132), and, if an intermediary server is used, the intermediary server reports the acceptance of the offer and the terms on which it was accepted to the offer-providing server. The offer-providing server then fulfills the offer by causing the offered product or service to be provided to the user (step 134).

DEPR:
Referring to FIG. 5, another network-based system for controlled transfer of information includes a client computer 200, a server computer 202 and an optional agency computer 204. Client computer 200 or agency computer 204 stores a client personal profile 206 containing demographic data, current shopping interests and preferences, contact addresses, and other personal or semi-personal information. The client personal profile can include information that changes on a day-to-day basis, such as a purchasing history (which may be recorded in accordance with the techniques described in the above-mentioned U.S. patent application Ser. No. 08/328,133), or a list of goods that the user wishes to buy (entered manually by the user in response to a prompt). Client computer 200 also stores a client security profile 208 that specifies that certain information in client personal profile 206 should be disclosed to server computer 202 only to trusted servers or only upon authorization from the client user or both. A client "avatar" 210 located at client computer 200 acts as an agent for the user by controlling the release of information from client personal profile 206 to server computer 202.

DEPR:
If the profile query requests information that the security profile restricts only to trusted servers, then the client avatar determines whether the server computer is one of the trusted servers and, if so, checks the authenticating signature contained in the offer/catalog description record (step 217) (the client avatar may assume that if the supplier of the record is a trusted supplier, then the server should be trusted too). If the profile query requests information that, according to the security profile, requires user authorization for release, then the client avatar prompts the user for authorization to release the information to the server computer (step 218) and the user indicates whether release of the information is authorized (step 220). Ordinarily, the user will not be prompted for authorization to release information to a trusted server, but the security profile can nevertheless be configured to require this for certain information.

DEPR:
After the client avatar determines which requested information can be released to the server computer, the client avatar transmits a subset of the client personal profile to the server computer, or sends an authorization message to the agency computer, which in turn transmits the subset of the client personal profile to the server computer (step 222). The subset includes all information in the client personal profile requested in the profile query and authorized for release to the server computer. Thus, the subset may not include all the information requested in the profile query. The server computer then transmits a client-specific sales offer or a customized document such as an electronic newspaper or magazine to the client computer based on the subset of the client personal profile received by the server computer (step 224), and the offer or document is displayed to the user at the client computer. The server computer may use the subset of the client personal profile to customize other web-based services offered to the user, including digital coupons, search services, and advertisements. Client-specific sales offers and coupons can be implemented in accordance with the smart digital offer technique described above in connection with FIGS. 3 and 4A-4B. The server computer could alternatively use the subset of the client personal profile to select or fabricate a channel object to send to the client computer, the channel object corresponding to a channel for asynchronous transfer of information to the client computer. The client computer can then activate the channel object in accordance with the

technique described above in connection with FIGS. 1 and 2. The server computer may even create a broadcast or multicast channel for the user by broadcasting or multicasting client-specific information and placing a specific identifying character or code at the beginning of the client-specific information. All of this can be accomplished using a single client personal profile stored at the client computer or agency computer, rather than multiple personal profiles stored at multiple server computers.

DEPR:
The security profile of the user can be developed progressively according to a scheme in which the security profile initially assumes that every supplier of offer/catalog description records is untrusted, every server is untrusted, and all information requires user authorization for release to every server. As profile queries are received by the client avatar, the client avatar queries the user whether the server computer should be trusted in the future (or whether the supplier of the offer/catalog description records should be trusted in the future, in which case the servers used by the trusted suppliers will be trusted too), and whether the requested information is authorized for release to untrusted servers. Based on the user's responses, the client avatar appropriately reconfigures the security profile.

DEPR:
In one embodiment, when the client avatar sends the subset of the client personal profile to the server computer, the client computer identifies the agency computer to the server computer. At the same time the client avatar sends an authorization message to the agency computer authorizing release of certain information, or any and all information, from the client personal profile to the server computer. This allows the server computer to transmit profile queries to the agency computer and to receive from the agency computer subsets of the client personal profile, even when the client computer is off-line. The agency computer maintains an access control list corresponding to all of the authorization messages received from the client computer, so that the agency computer can know which information can be released to which servers.

CLPR:
48. The network-based system of claim 47 wherein the client computer is programmed to record in the metering log mouse-click activity on the portion of the document corresponding to the embedded link and to allow the mouse-click activity to pass on to objects on the document other than the embedded link.

CLPR:
59. The network-based system of claim 58 wherein the client computer is programmed to record in the metering log mouse-click activity on the portion of the document corresponding to the embedded link and to allow the mouse-click activity to pass on to objects on the document other than the embedded link.

CLPV:
an agency computer programmed to store the personal profile, wherein the client avatar causes an authorization message to be transmitted to the agency computer authorizing the agency computer to release the subset of the personal profile, and the agency computer is programmed to transmit the subset of the personal profile to the server computer.

CLPV:
wherein the embedded link is structured to require the client computer to search for information stored on the client computer pertaining to authorization of the user activating the embedded link.

WEST

Generate Collection

L12: Entry 2 of 6 File: USPT May 8, 2001

US-PAT-NO: 6230173
DOCUMENT-IDENTIFIER: US 6230173 B1

TITLE: Method for creating structured documents in a publishing system

DATE-ISSUED: May 8, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Ferrel; Patrick J.	Seattle	WA		
Meyer; Robert F.	Redmond	WA		
Millet; Stephen J.	Seattle	WA		
Shewchuk; John P.	Seattle	WA		
Smith; Walter W.	Seattle	WA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Microsoft Corporation	Redmond	WA			02

APPL-NO: 8/ 503307
DATE FILED: July 17, 1995

INT-CL: [7] G06F 17/30
US-CL-ISSUED: 707/513; 707/501
US-CL-CURRENT: 707/513; 707/501.1
FIELD-OF-SEARCH: 395/774, 395/776-778, 707/513, 707/501, 707/514, 707/515

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4710885	December 1987	Litteken	395/774
☐	4969093	November 1990	Barker et al.	395/800
☐	5347632	September 1994	Filepp et al.	395/200.09
☐	5475805	December 1995	Murata et al.	395/774
☐	5557722	September 1996	DeRose et al.	395/774

OTHER PUBLICATIONS

Duncan, Ray, "Power Programming: An HTML Primer," PC Magazine, Jun. 13, 1995, pp. 261-270.
Sperberg-McQueen et al., "HTML to the Max: a Manifesto for Adding SGML Intelligence to the World-Wide Web",

http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/Autools/sperberg-McQueen/sperberg.html., Oct. 1994.

ART-UNIT: 216
PRIMARY-EXAMINER: Hong; Stephen S.
ATTY-AGENT-FIRM: Banner & Wilcoff, Ltd.

ABSTRACT:

An authoring environment for producing content for an on-line system is described. This environment includes a story editor which can save files in a Multimedia Document Format (MDF) file. A MDF file is an OLE storage wherein one storage object holds text of the content in a Multimedia Publishing Markup Language. Other parts of the MDF file include storages for holding content search terms and storages for embedded objects.

15 Claims, 19 Drawing figures

L12: Entry 2 of 6 File: USPT May 8, 2001

DOCUMENT-IDENTIFIER: US 6230173 B1
TITLE: Method for creating structured documents in a publishing system

ABPL:
An authoring environment for producing content for an on-line system is described. This environment includes a story editor which can save files in a Multimedia Document Format (MDF) file. A MDF file is an OLE storage wherein one storage object holds text of the content in a Multimedia Publishing Markup Language. Other parts of the MDF file include storages for holding content search terms and storages for embedded objects.

BSPR:
The present invention relates to electronic publishing systems and, more specifically, to an authoring system for creating structured documents in an on-line publishing system.

BSPR:
Therefore a need exists for an on-line system which provides separation of design from content. Moreover, a need exists for an authoring system to be used in an on-line network to provide content providers with increased flexibility for presenting their content to customers.

BSPR:
The present invention relates to a new authoring system for creating on-line stories. The preferred embodiment of the environment uses an enhanced version of Microsoft Word.RTM. to create Multimedia Document Files (MDF). These multimedia files are then used to provide content for displayed on-line titles as discussed below for a Multimedia Publishing System (MPS).

BSPR:
In addition to adding MDF content to a project by authoring in Word.RTM., the present invention also includes programs for converting existing HTML documents to a MPML when added to a project. These concepts will be explained in more detail below.

DEPR:
Reference is now made to the drawings wherein like numerals refer to like parts throughout. For convenience, the following description will be organized into the following seven principle sections: Acronyms, Advantages of the Multimedia Publication System, Multimedia Publishing System Overview, Authoring Overview, Multimedia Document Format File Structure, Using Multimedia Documents in an On-line System, Summary.

DEPR:
To enable a new generation of on-line, multimedia applications, an end-to-end system has been invented for developing and using applications and services. The system, called the Multimedia Publishing System (MPS or MP system), preferably uses the Microsoft Network. As an open, turnkey system, MPS includes components for design, authoring, distribution, viewing, search, personalization, and billing of on-line services and multimedia applications. The MP system allows content providers to offer rich, interactive multimedia applications and services, providing users a compelling and exciting on-line experience. The MP system provides the key to overcoming the previously described hurdles facing the on-line industry.

DEPR:

The Designer 194 is an extensible design and development environment that includes several preferred software components. These include a project editor 184 to manage tiles, containers, and objects; a page editor 186 to create and layout pages; a word processor, such as Microsoft MPS Word, for creating content optimized for the MP system 100; and optional third-party tools, such as a sound editor 190, an image editor 192, and another media object editor 193 to create and modify sound, image, video, animation and other content objects. For authoring textual content, the preferred text editor is an enhanced version of the Microsoft Word 6.0 wordprocessing program for creating tagged, hypertext documents. Together, these programs form the Designer Component 194.

DEPR:
The MPS Designer 194 is a page or forms-based development system similar to Visual Basic. The development environment is graphical and easy to use. Controls, which represent the components of a MPS application that will appear on-screen, are laid out within MPS pages. MPS pages and controls are preferably based on Object Linking and Embedding 198 (in FIG. 2) (OLE), Microsoft's component software technology. OLE, which presently is at version 2, is further described in Inside OLE 2 and OLE 2 Programmer's Reference, Volumes 1 and 2, all of which are published by Microsoft Press, and are hereby incorporated by reference. However, other compound document architectures such as OpenDoc could be used as well.

DEPR:
While content is displayed within controls that have been laid out on MPS pages in the MPS Designer 194, content can be authored in any number of existing Microsoft and third-party tools. One such tool for authoring hypertext is an enhanced version of Microsoft Word that supports special MPS features for creating and tagging MPS text. Other existing tools for creating bitmaps, complex drawings, and other multimedia content can be used to create the content displayed within any particular OLE Control. In addition, most existing OLE Controls (.ocx executable programs) will work in the MPS environment although they may not be optimized for on-line applications. For example, a standard AVI OLE Control could be placed in an MPS application.

DEPR:
For dynamic titles, most (and potentially all) of the work is done within the Content Authoring environment. For static titles, it could all be done within the Title Design environment. In practice, most releases will involve some work in both of these environments.

DEPR:
Content authors--including editors, writers, reporters, and forum managers--generate content, including structured stories, using the content authoring environment. Writers compose the textual content that appears in a title (or a release of a title). They hand their materials off to the editorial staff. The editorial staff is in charge of the overall content of the title. For multimedia titles, this role is very similar to the director of a motion picture or television program.

DEPR:
The content authoring environment supports a variety of tools, such as, for example, a MPS document editor. The MP system 100 also supplies tools to specify and manage links and to specify story properties. Third-party tools may also be added to the content authoring environment.

DEPR:
The present invention includes a set of authoring tools and data structures for creating content that is to be published in an on-line network. The present invention includes a story editor which is used by the publisher 102 to produce content for an on-line publishing system. The preferred embodiment of the invention uses an enhanced version of Microsoft Word.RTM. to create Multimedia Document Format (MDF) files. The enhanced version of Microsoft Word is also known as MPS Word. These MDF files are then used to provide content for displayed on-line titles as discussed below for the Multimedia Publishing System (MPS).

DEPR:

In addition to adding MDF content to a project by authoring in Word, converting existing HTML documents to MPML when added to a project is also supported. These concepts will be explained in more detail below.

DEPR:
Once MDF files are added to a project as content, they cannot be directly edited in Word since Word cannot directly read an OLE compound file such as in MDF format. If the user wishes to edit a document that has been converted to a MDF file, it must first be exported from the project to a temporary file. The project then launches the enhanced Word, and tells it to open the temporary file. At this point, Word will use the MPML input converters to read the file and save changes. When the edit operation is complete, the project must be notified to read back in the changes from the temporary file. This is accomplished by overriding the Word Save command with macros provided in the template used for authoring MP system content.

DEPR:
Briefly, the drag and drop capability allows an author to drag an icon representing an embedded object from the object editor and drop it within the document. By simply dragging and dropping the object within the document, a link is established from that document to the embedded object. Upon saving, the embedded object 565 is saved in a storage and stream below the root IStorage of the MDF file 521. The protocols and procedures for setting up IStorages and IStreams in a compound structured OLE document are well known. However, the structure of the document shown in FIG. 10 provides significant advantages over prior on-line authoring systems wherein only tagged text could be used in the on-line system. Other advantages of the document structure shown in FIG. 10 will become more apparent in reference to the following figures.

DEPR:
After selecting the appropriate object tag at state 670, the tags are applied to the object at a state 672. Following application of the tags, the process 582 loops back to the decision state 654 wherein the system 582 queries whether to insert text into the story. If there is no text to be inserted into the story at decision state 654 and no embedded object to be inserted into the story at decision state 664, then the process 582 moves to a decision state 674 wherein a query is made whether to insert a hypertext link into the story.

DEPR:
After the importance of the linked object has been selected at state 692, the link editor dialog is closed at a state 694 wherein the process 582 then moves back to the decision state 654 wherein the process 582 queries whether to insert text into the story. If there is no text to be inserted into the story at decision state 654 and no embedded object to be inserted at decision state 664 and no hypertext link to be embedded at decision state 674 then a decision is made at a decision state 694 whether to add find properties to the document.

DEPR:
If the current tag is an embedded object tag at decision state 810, then a link pointer to the object is placed in the text at state 812 using the name of the entity as a reference. The embedded object is then saved to an object storage which has a data stream and results stream. Saving an object within a storage is well known within the OLE structured storage system. Once the object has been saved into an object storage at state 814, a bitmap is saved to the results stream of the object storage at state 816.

DEPR:
However, if a decision is made at decision state 986 that the embedded object did have a wrap style set at state 970, the process 615 positions the object to the correct place in the control region at a state 990. The position that the embedded object takes within the control region at state 990 is determined by referencing the style that was set at state 970 to the linked style sheet. For example, if the set style was "wrap-advertisement", and upon referencing the style sheet the control determined that this style means to place the embedded object in the upper, right corner of the control region, the object will therefore be appropriately placed at state 990.

DEPR:
As stated above, the styles contained in every style sheet are predefined by the MP system authoring program. In a presently preferred embodiment, this program is a version of the Microsoft Word.RTM. program, termed MPS Word, that has the special capability of producing documents formatted in a Multimedia Document Format that was described in detail in reference to FIG. 10. A part of the MDF is a new markup language known as MPML which is a form of an SGML. However, MPML has formatting commands unique to the MP system. Markup languages which are well known in on-line networks identify portions of documents by embedded tags. In an MPML document, there is one MPML tag per document portion and each tag is mapped to a style that is found in a style sheet.

DEPL:
Authoring and Title Release

DEPL:
Title This is the titleSubject This is the subjectAuthor George WashingtonKeywords Authoring, Word, MultimediaPriority 5

DEPC:
IV. AUTHORING OVERVIEW

WEST

Generate Collection

A

L10: Entry 1 of 3 File: PGPB Nov 29, 2001

PGPUB-DOCUMENT-NUMBER: 20010047387
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20010047387 A1

TITLE: Systems and methods for providing distributed cross-enterprise portals

PUBLICATION-DATE: November 29, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Brockhurst, Russell A.	Austin	TX	US	

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	COUNTRY	TYPE CODE
Exoplex, Inc.				02

APPL-NO: 09/ 811209
DATE FILED: March 16, 2001

RELATED-US-APPL-DATA:

RLAN	RLFD	RLPC	RLKC	RLAC
60192729	Mar 27, 2000			US
60213384	Jun 23, 2000			US

INT-CL: [07] G06F 15/16, G06F 9/44

US-CL-PUBLISHED: 709/203; 709/315
US-CL-CURRENT: ~~709/203; 709/315~~

REPRESENTATIVE-FIGURE: 4

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This application claims priority from U.S. Provisional Patent Application Ser. No. 60/192,729, filed Mar. 27, 2000, and entitled "SYSTEM AND METHOD FOR PROVIDING DISTRIBUTED CROSS-ENTERPRISE PORTALS" and U.S. Provisional Patent Application Ser. No. 60/213,384, filed Jun. 23, 2000, and entitled "SYSTEM AND METHOD FOR PROVIDING DISTRIBUTED CROSS-ENTERPRISE PORTALS."

ABSTRACT:

A system and method for providing distributed cross-enterprise portals is disclosed. In one form, a distributed portal to portal system includes at least one supplier portal operable to provide communication between a plurality of networks and at least one encapsulated component operably associated with the at least one supplier portal. The system further includes at least one user portal operable to receive a plurality of distributed encapsulated components. The encapsulated components may further include a global identifier operable to globally identify the distributed component.

WEST

Generate Collection

L10: Entry 1 of 3 File: PGPB Nov 29, 2001

DOCUMENT-IDENTIFIER: US 20010047387 A1
TITLE: Systems and methods for providing distributed cross-enterprise portals

DETX:
[0039] In one embodiment, network system 100 may employ encapsulated components having a standard operating environment and standard external interfaces (methods, transactions, and queries), offer a web based user interface (HTML; etc.), and be secure from intrusion from un-authorized access. The standard operating environment (not expressly shown) may include web hosting services (http, ftp, active components, scripts, etc.), database support (SQL Server, etc.), and middleware services (CORBA, DCOM, etc.). Regardless of where a component physically resides, a supplier or publisher may retain ownership and control of the encapsulated component creating a trusted extension of the publisher's internal information system.

DETX:
[0045] Common portal components 202 may be accessible to both administrators and end users through a web based user interface (not expressly shown). Authorized administrators will be able to use this interface to manage security for the cross-enterprise portal 200 including the administration of component interfaces 204, 205, 206. End users may use common portal component 202 to search for other portal components and to access general news. For example, a news item could be posted to common portal component 202 each time a new service (component) is installed.

DETX:
[0046] In one embodiment, each component 208, 209, 210 and 211 may interact through standard interfaces (transactions or queries). For example, each component may provide end users with intranet user interface 205 (accessible through a standard web browser). Additionally, each interface 204, 205, 206 may be used by authorized administrators to set configuration options for components 208, 209, 210 and 211 and by end users to access services offered by components 208, 209, 210 and 211. As illustrated, supplier components 209 may be used to establish secure connections to the appropriate backend systems using secure supplier channels via Internet 212. Legacy front-end components 210 may establish connections to legacy systems through an associated intranet 207.

DETX:
[0060] In one embodiment, external interface 401 or 402 could allow objects 405 and 407 to store persistent states 406 and 408 across component boundaries. For example, encapsulated component A 403 or B 404 may include a compound document having persistent state information saved by objects initiated in another component. This would not violate the integrity of encapsulated components A 403 or B 404 as long as the persistent state information were not used to initialize new objects in the context of different components. In one embodiment, the structure of existing compound documents could be extended to store component context information along with other state information. Additionally, object oriented databases could also be modified to store component state information. By storing component state information with other state information, an operating system could set the correct component context for context switch 410 during object initialization.

DETX:
[0062] An encapsulated component may be operable inside an internal firewall.

In one embodiment, the system resources required by the component may be included within the firewall such as files, databases, executable modules, configuration data, and back-end connections (Open Database Connectivity ("ODBC") links, middleware, etc.). The component may use these resources freely and can be isolated from using resources associated with other components unless authorized by the other component's publisher. These internal firewalls improve the stability of the cross-enterprise portal by minimizing unwanted component interaction.

DETX:
[0081] Retail mall 601, 606 and/or 611 may include a collection of "stores" and "aggregators" associated with a specific distribution channel or channels. For example, an aggregator component (not expressly shown) may be associated with mall B606 and may include a collection of stores that offer similar products and/or services. Aggregator components may implement marketplaces for specific product categories. Retail Mall 600 may include utilities components (not expressly shown) having common services that may be used by other components of a retail mall 601, 606 and 611 such as session management, shopping cart management, credit card authorization, map generation, and advertising. Utilities components may be developed and distributed by software vendors and systems integrators.

WEST

Generate Collection

L10: Entry 2 of 3 File: PGPB Nov 1, 2001

PGPUB-DOCUMENT-NUMBER: 20010037467
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20010037467 A1

TITLE: Controlled transfer of information in computer networks

PUBLICATION-DATE: November 1, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
O'Toole, James W. JR.	Cambridge	MA	US	
Gifford, David K.	Weston	MA	US	

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	COUNTRY	TYPE CODE
Open Market, Inc.				02

APPL-NO: 09/ 897407
DATE FILED: July 3, 2001

RELATED-US-APPL-DATA:

RLAN	RLFD	RLPC	RLKC	RLAC
09897407	Jul 3, 2001	GRANTED	A1	US
08741862	Oct 29, 1996	PENDING	A1	US
6279112	Jul 3, 2001	GRANTED		US
09897407	Nov 14, 2000	GRANTED		US
09711511	Nov 14, 2000			US
09711511	Mar 2, 1998			US
09033143	Mar 2, 1998			US
6195649	Nov 29, 1995			US
09033143				US
08563745				
5724424				

INT-CL: [07] G06F 11/30

US-CL-PUBLISHED: 713/201
US-CL-CURRENT: 713/201

REPRESENTATIVE-FIGURE: 1

ABSTRACT:

The present invention relates to techniques for controlling transfers of information in computer networks. One technique involves transmitting from a server computer to a client computer a document containing a channel object corresponding to a communication service, and storing an access ticket that indicates that a user of the client computer permits the information source computer to communicate with the user over a specified channel. Another technique involves transmitting smart digital offers based on information such

technique involves transmitting smart digital offers based on information such as coupons and purchasing histories stored at the computer receiving the offer. Another technique involves transmitting from a server computer to a client computer a request for a user's personal profile information, and activating a client avatar that compares the request for personal profile information with a security profile of the user limiting access to personal profile information. Another technique involves transmitting from a server computer to a client computer a document containing an embedded link, activating the embedded link at the client computer and recording activation of the embedded link in a metering log.

L10: Entry 2 of 3 File: PGPB Nov 1, 2001

DOCUMENT-IDENTIFIER: US 20010037467 A1
TITLE: Controlled transfer of information in computer networks

BSTX:
[0003] U.S. patent application Ser. No. 08/168,519, filed Dec. 16, 1993 by David K. Gifford and entitled "Digital Active Advertising," the entire disclosure of which is hereby incorporated herein in its entirety by reference, describes a network sales or payment system that includes at least a client computer and a payment computer. The client computer transmits a payment order and an authenticator to the payment computer. The payment computer verifies the authenticator, transmits a payment authorization message and an authenticator back to the client computer, and performs a payment settlement transaction.

BSTX:
[0012] The client avatar acts as an agent for the user by controlling the release of information from the client personal profile to the server computer. The client avatar makes it possible to store a single client personal profile at the client computer or an agency computer, rather than multiple personal profiles at multiple server computers, while at the same time limiting the release of certain information from the personal profile only to trusted servers or only upon specific authorization from the user.

DETX:
[0024] Referring to FIG. 1, a network-based system for controlled asynchronous transfer of information includes a client computer 10, operated by a user, that filters information transferred asynchronously to the client computer, a server computer 12 that transmits a document to the client computer containing a channel object that can be activated to authorize an asynchronous transfer of information, an information source computer 14 that asynchronously transfers the information, and an optional notification server 16 that acts as a trusted intermediary that filters asynchronously transferred information on behalf of the client computer. In certain implementations server computer 12 and information source computer 14 are the same computer. As used herein, the term "asynchronous" transfer of information refers to a transfer of information from an information source computer that is initiated by the information source computer rather than by another computer to which the information source computer responds.

DETX:
[0027] The description of the asynchronous communication service in the channel object may include a certificate that includes an identification of the supplier of the information to be transmitted to the client computer, as well as the supplier's public key, the certificate being signed by a certifying authority. This public key will be used by the client computer to authenticate the information to be transmitted to the client computer by the information source computer.

DETX:
[0029] When the document is displayed on the user computer, the icon contained in the channel object is displayed on the document as a representation of the channel object, and the user can determine from the document whether to authorize delivery of the content of the channel object as described in the document. The user can activate or select the channel object by clicking on a representation of the channel object on the document, or a channel object in a document or broadcast received by the client computer may be activated

automatically by the computer if the keywords or the other identifying information contained in the channel object match preset parameters pre-programmed into the client computer as a personal profile of the user (step 28). For example, the user may pre-program the computer to search for a keyword phrase such as "BUGS BUNNY" to automatically activate channel objects pertaining to BUGS BUNNY. similarly, the user may authorize automatic activation of channel objects containing an embedded "G" rating, or automatic activation of only one megabyte of information per week.

DETX:
[0030] Activation of the channel object causes an access ticket containing the description of the asynchronous communication service to be added to the client control list in the client computer, or causes the access ticket to be sent to the notification server, which adds it to the access control list (step 30). The access ticket permits the information source computer to communicate asynchronously with the client computer over a channel specified by the channel object, which may be a broadcast or multicast channel at a specific time period, or which may be the computer network linking the client computer and the information source computer in the event that the information from the information source computer is to be received by means of an asynchronous communication over the computer network. Thus, the activation of a channel object initiates an asynchronous communication channel from the information source computer to the client computer and instructs the client computer that the information source computer is authorized to send information over the channel.

DETX:
[0038] Channel objects may be embedded not only in documents or pages on the World Wide Web, but in an alternative implementation they may be embedded in e-mail messages, OLE objects, ActiveX applets, etc. In fact, all of the communications between the server computer and the client computer and between the information source computer and the client computer may occur by e-mail, via compound documents, etc.

DETX:
[0044] If the smart digital offer object attempts to observe the purchasing history or certain other user-specific information, the client computer asks the user whether the user wishes to reveal the information (step 122). The user indicates whether release of the information is authorized (step 124), and the smart digital offer object then examines the coupon (including the coupon's authenticator), digital receipts (including authenticators) and other user-specific information authorized to be revealed by the user, and presents to the user an offer of a product or service (step 126). The execution environment at the client computer can under some circumstances change between steps 118 and 126. For example, the client computer may receive a coupon after step 118 occurs but before step 126 occurs. In one particular embodiment the client computer includes a client "avatar" of the type described below in connection with FIGS. 5 and 6, which limits the release of certain information only to trusted servers, or only upon authorization from the client user, or both.

DETX:
[0045] The terms or conditions of the offer, such as price and payment terms, are calculated by the smart digital offer object using formulas that depend on the information contained in the digital coupons and the other information examined by the smart digital offer object, including the time of day, or user profile information such as membership codes, user's age, user's income, and other demographic information certified by an independent authority with an authenticator. When the user accepts the offer (step 128) the client computer sends a message to the offer-providing server indicating that the user has accepted the offer, or sends the message to an intermediary server that is trusted by the client computer to maintain the confidentiality of user-specific information and is trusted by the offer-providing server to verify the terms on which the offer was accepted (step 130). The message sent to the offer-providing server or the intermediary server includes the terms upon which the offer was accepted and also includes an authenticator. The offer-providing server or the intermediary server verifies the terms on which the offer was accepted by verifying the authenticator (step 132), and, if an intermediary server is used, the intermediary server reports the acceptance of

the offer and the terms on which it was accepted to the offer-providing server. The offer-providing server then fulfills the offer by causing the offered product or service to be provided to the user (step 134).

DETX:
[0052] Referring to FIG. 5, another network-based system for controlled transfer of information includes a client computer 200, a server computer 202 and an optional agency computer 204. Client computer 200 or agency computer 204 stores a client personal profile 206 containing demographic data, current shopping interests and preferences, contact addresses, and other personal or semi-personal information. The client personal profile can include information that changes on a day-to-day basis, such as a purchasing history (which may be recorded in accordance with the techniques described in the above-mentioned U.S. patent application Ser. No. 08/08/328,133), or a list of goods that the user wishes to buy-(entered manually by the user in response to a prompt). Client computer 200 also stores a client security profile 208 that specifies that certain information in client personal profile 206 should be disclosed to server computer 202 only to trusted servers or only upon authorization from the client user or both. A client "avatar" 210 located at client computer 200 acts as an agent for the user by controlling the release of information from client personal profile 206 to server computer 202.

DETX:
[0054] If the profile query requests information that the security profile restricts only to trusted servers, then the client avatar determines whether the server computer is one of the trusted servers and, if so, checks the authenticating signature contained in the offer/catalog description record (step 217) (the client avatar may assume that if the supplier of the record is a trusted supplier, then the server should be trusted too). If the profile query requests information that, according to the security profile, requires user authorization for release, then the client avatar prompts the user for authorization to release the information to the server computer (step 218) and the user indicates whether release of the information is authorized (step 220). Ordinarily, the user will not be prompted for authorization to release information to a trusted server, but the security profile can nevertheless be configured to require this for certain information.

DETX:
[0055] After the client avatar determines which requested information can be released to the server computer, the client avatar transmits a subset of the client personal profile to the server computer, or sends an authorization message to the agency computer, which in turn transmits the subset of the client personal profile to the server computer (step 222). The subset includes all information in the client personal profile requested in the profile query and authorized for release to the server computer. Thus, the subset may not include all the information requested in the profile query. The server computer then transmits a client-specific sales offer or a customized document such as an electronic newspaper or magazine to the client computer based on the subset of the client personal profile received by the server computer (step 224), and the offer or document is displayed to the user at the client computer. The server computer may use the subset of the client personal profile to customize other web-based services offered to the user, including digital coupons, search services, and advertisements. Client-specific sales offers and coupons can be implemented in accordance with the smart digital offer technique described above in connection with FIGS. 3 and 4A-4B. The server computer could alternatively use the subset of the client personal profile to select or fabricate a channel object to send to the client computer, the channel object corresponding to a channel for asynchronous transfer of information to the client computer. The client computer can then activate the channel object in accordance with the technique described above in connection with FIGS. 1 and 2. The server computer may even create a broadcast or multicast channel for the user by broadcasting or multicasting client-specific information and placing a specific identifying character or code at the beginning of the client-specific information. All of this can be accomplished using a single client personal profile stored at the client computer or agency computer, rather than multiple personal profiles stored at multiple server computers.

DETX:

[0056] The security profile of the user can be developed progressively according to a scheme in which the security profile initially assumes that every supplier of offer/catalog description records is untrusted, every server is untrusted, and all information requires user authorization for release to every server. As profile queries are received by the client avatar, the client avatar queries the user whether the server computer should be trusted in the future (or whether the supplier of the offer/catalog description records should be trusted in the future, in which case the servers used by the trusted suppliers will be trusted too), and whether the requested information is authorized for release to untrusted servers. Based on the user's responses, the client avatar appropriately reconfigures the security profile.

DETX:
[0057] In one embodiment, when the client avatar sends the subset of the-client personal profile to the server computer, the client computer identifies the agency computer to the server computer. At the same time the client avatar sends an authorization message to the agency computer authorizing release of certain information, or any and all information, from the client personal profile to the server computer. This allows the server computer to transmit profile queries to the agency computer and to receive from the agency computer subsets of the client personal profile, even when the client computer is off-line. The agency computer maintains an access control list corresponding to all of the authorization messages received from the client computer, so that the agency computer can know which information can be released to which servers.

CLTX:
23. The network-based system of claim 22 wherein the client computer is programmed to ask the user whether the user wishes to reveal the user profile information and the client computer releases the user profile information for use by the smart digital offer only if the user authorizes release of the user profile information.

CLTX:
36. The network-based system of claim 35 further comprising an agency computer programmed to store the personal profile, wherein the client avatar causes an authorization message to be transmitted to the agency computer authorizing the agency computer to release the subset of the personal profile, and the agency computer is programmed to transmit the subset of the personal profile to the server computer.

CLTX:
55. The network-based system of claim 53 wherein the embedded link is structured to require the client computer to search for information stored on the client computer pertaining to authorization of the user activate the embedded link.

WEST

End of Result Set

Generate Collection

L10: Entry 3 of 3 File: PGPB Aug 30, 2001

PGPUB-DOCUMENT-NUMBER: 20010018739
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20010018739 A1

TITLE: Method and system for processing electronic documents

PUBLICATION-DATE: August 30, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Anderson, Milton	Fair Haven	NJ	US	
Jaffe, Frank	Boston	MA	US	
Hibbert, Chris	Los Altos	CA	US	
Virkki, Jyri	Scott Vly	CA	US	
Kravitz, Jeffrey	Yorktown Heights	NY	US	
Chang, Sheveling	Cupertino	CA	US	
Palmer, Elaine	Goldens Bridge	NY	US	

APPL-NO: 09/ 750379
DATE FILED: December 28, 2000

RELATED-US-APPL-DATA:

RLAN	RLFD	RLPC	RLKC	RLAC
09750379	Dec 28, 2000	GRANTED	A1	US
09386551	Aug 31, 1999	GRANTED		US
6209095	Aug 31, 1999			US
09386551	Dec 19, 1997			US
08994636	Dec 20, 1996			US
6021202				US
60033896				

INT-CL: [07] H04L 9/30, H04L 9/32

US-CL-PUBLISHED: 713/176; 380/243; 705/75
US-CL-CURRENT: 713/176; 380/243; 705/75

REPRESENTATIVE-FIGURE: 6

ABSTRACT:

The invention includes a markup language according to the SGML standard in which document type definitions are created under which electronic documents are divided into blocks that are associated with logical fields that are specific to the type of block. Each of many different types of electronic documents can have a record mapping to a particular environment, such as a legacy environment of a banking network, a hospital's computer environment for electronic record keeping, a lending institution's computer environment for processing loan applications, or a court or arbitrator's computer system. Semantic document type definitions for various electronic document types (including, for example, electronic checks, mortgage applications, medical

(including, for example, electronic checks, mortgage applications, medical records, prescriptions, contracts, and the like) can be formed using mapping techniques between the logical content of the document and the block that is defined to include such content. Also, the various document types are preferably defined to satisfy existing customs, protocols and legal rules.

WEST

End of Result Set

Generate Collection

L10: Entry 3 of 3 File: PGPB Aug 30, 2001

DOCUMENT-IDENTIFIER: US 20010018739 A1
TITLE: Method and system for processing electronic documents

BSTX:
[0009] In addition to the inherent inefficiencies of paper transactions, other problems exist. Many of these problems relate to documents that require signatures. In particular, in order for a reader of a paper document to determine that a particular document or part of a document has been signed, the reader must be given access to the entire document; thus, a party who may only need to know that the document has been signed must be given access to the entire document, including any confidential information contained therein. Signatures are used in a wide range of contexts, including financial instruments, contracts, mortgage applications, and medical records and prescriptions, to indicate the agreement, consent or authority of the signer. Transactions that require signatures have traditionally employed conventional means for execution, such as pen and paper. As used herein, "signature" has its broadest source; that is, it means any indication of agreement, consent, certification, acceptance, or other giving of authority, that is associated with a person or entity.

BSTX:
[0037] Processing a paper check requires time as the physical check is routed to the payer, the payee, the payee's bank, the clearing house and/or the payer's bank. The same is true of other types of financial transactions involving paper instruments, such as credit card slips generated during a credit card sale. In a credit card transaction, a merchant makes an impression of the customer's card, which the customer then signs, to function as a receipt for the transaction. The merchant typically obtains a positive acknowledgment or credit authorization from the customer's credit card company before accepting the credit card slip. This assures that payment will be received.

BSTX:
[0052] Mortgage loan transactions raise similar confidentiality concerns as legal contracts. A credit reporting agency may only need to see the part of the application that authorizes a credit report, but known electronic techniques require the signer to sign the entire file; thus, in order to ensure the validity of the signature, the credit reporting agency must receive the entire document. Other third parties may also need to see only part of the application. Accordingly, a need has arisen to provide for transmission of part of a mortgage loan application while ensuring the integrity and validity of the signature, as well as of the information in the part that is transmitted.

BSTX:
[0060] In one embodiment, the invention features a computer-based method in which an electronic instrument is created for effecting a transfer of funds from an account of a payer in a funds-holding institution to a payee, the instrument including an electronic signature of the payer. A digital representation of a verifiable certificate by the institution of the authenticity of the account, the payer, and the public key of the payer is appended to the instrument. This enables a party receiving the instrument, e.g., the payee or a bank, to verify the payer's signature on the instrument. A similar certificate of authenticity could also be issued in other contexts. For example, a certifying authority could certify that a doctor is properly

licensed and authorized to sign a prescription. A certifying authority could certify as to the creditworthiness of a borrower in a transaction. A certifying authority could certify as to the authority of an individual to sign a contract for a given company. These examples are merely illustrative of all transactions in which a certifying entity participates.

BSTX:
[0062] Also appended to the electronic document may be digital representations of a verifiable signature of a second signer. The second signer may be the payee of an electronic check, a second or doctor, a mortgage lender, for example. A verifiable certificate by a third party, such as an institution which holds an account of the payee of an electronic check, or a credit institution in the case of a mortgage application, may also be appended, as may be a verifiable certificate by a central authority, such as a banking authority, with respect to the third party, such as the institution which holds the payee's account in the case of the electronic check.

BSTX:
[0069] Implementations of the invention may include one or more of the following features. The memory may contain certification information provided by the institution and which is usable to append secure, verifiable certificates to electronic documents to certify a relationship between an owner of the signature and a public key of the owner. A unique identifier may be assigned to each electronic document. The portable token may be a PCMCIA compatible card, smart card or smart disk, which may internally hold a private signature key and a secure memory for the check serial number. The certification information may be given a limited useful life. The memory may also contain certification information provided by a third party authority, such as a central banking authority in the case of an electronic check, and which is usable to append secure, verifiable certificates to electronic documents to certify the authenticity of a party, such as the funds-holding institution in the case of the electronic check. The certification information provided by the third party authority may have a limited useful life. In the electronic check embodiment of the present invention, the central banking authority may be a United States Federal Reserve Bank. The memory may also contain a complete or partial register of electronic documents, or a subset of the information contained in the documents, to which signatures have been appended. The appended signature may be a signature of any party to a transaction, such as a payer who holds the account in the institution, an endorsement signature of a payee, a signature of a doctor or patient, a signature of a borrower, broker or lender, or the signature of a contracting party. The memory may also contain a personal identification number for controlling access to the memory.

DETX:
[0159] Referring to FIG. 36, whenever a block 804 is to be authenticated, or tamper-proofed, a digital signature block 820 is added to the electronic document. The signature block 820 contains a reference to a certificate block 822 containing a public key used to verify the digital signature. The signature block can also be used to bind multiple blocks together, so that the resulting compound document can be verified. The FSML tags for signature blocks in an embodiment of the invention are depicted in FIG. 40.

DETX:
[0160] When combining an FSML block into a larger, compound document, the names of the original blocks may not be unique. As such, the document combining process also operates to handle naming conflicts when the documents being combined use block names that are not unique. FIG. 41 depicts the FSML tags for combining blocks.

DETX:
[0184] In any embodiment, once the template is filled in by the payer as the FSML document is complete, the electronic check may be signed by passing it through the payer's electronic checkbook. The electronic checkbook is contained within a PCMCIA card containing the payer's private signature key and certificates from the bank and the federal reserve. The certificates may be cryptographically signed letters of reference attesting to the validity of the payer's account and the payer's authority to write checks against the account, and the bank, respectively.

DETX:
[0186] Whenever a block must be authenticated, or tamper-proofed, a digital signature block is added to the electronic document. The signature block contains a reference to a certificate block containing a public key used to verify the digital signature. The signature block can also be used to bind multiple blocks together, so that the resulting compound document can be verified.

DETX:
[0187] When combining FSML into a larger, compound document, the names of the original blocks may not be unique. As such, the document combining process also operates to handle naming conflicts when the documents being combined use block names that are not unique.

DETX:
[0191] A public key, applied to verify cryptographic digital signature, is always generated in conjunction with the private key which is used to create the signature. The payer's digital signature 126, the payer's public verification key 134, and the message which was signed are used as inputs to the public key signature verification algorithm, which produces a true or false value. Public key cryptographic signatures are useful because the signature of a signer, computed using the signer's private key, can be verified by anyone else who knows the signer's public key. Since the signer computes his signature on a document using his private key, and since the verifier verifies the signer's signature using the signer's public key, there must be a way for the verifier to trust the association between the signer (and his account information) and the public key used to verify the signer's signature on the electronic check. Cryptographic signatures are used to sign checks when they are written, co-signed, endorsed and processed. Cryptographic signatures are also used by certification authorities to sign certificates or "letters of reference" that contain a name or description of a signer and the signer's public key. Thus, anyone who trusts the certification authority and who knows the certification authority's widely publicized signature verification key can verify the certificate and trust the signer's public key for use in verifying the signer's signature.

DETX:
[0220] The electronic checkbook is issued by the bank that holds the electronic checking account. Initialized electronic checkbooks may be sent to the account holder, in which case the PIN should be sent separately for security reasons. Alternatively, uninitialized cards may be distributed to bank branches. The bank officer can then use a trusted initialization terminal and a special smart card identifying the bank officer to establish a secure connection to a centralized CIS. The new card is inserted into the terminal to be initialized. This method has the advantage of making electronic checkbooks immediately available to new customers, accounts can be added to electronic checkbooks already being used by the customer, and certificates can be refreshed prior to their expiration dates without issuing new electronic checkbooks. The bank, or its agent, is also acting as a certifying authority since it is responsible for authenticating the identity of the electronic checkbook and PIN are delivered to the correct person. The electronic check may also support correspondent banking relationships, and will allow another bank or approved third party to act as a stand-in processor for electronic checks for banks that are unable to directly support the processing requirements for electronic checks. This will facilitate electronic check deployment in a secure way without affecting the traditional bank-customer relationship.

WEST

Help | Logout | Interrupt

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers | Preferences

Search Results -

Terms	Documents
(authori$ and (compoun$ adj document$))	3

Database: US Patents Full-Text Database
US Pre-Grant Publication Full-Text Database
JPO Abstracts Database
EPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Refine Search: (authori$ and (compoun$ adj document$)) Clear

Search History

Today's Date: 1/16/2002

WEST

Generate Collection

hit 3/11/03

L12: Entry 3 of 6 File: USPT Nov 2, 1999

US-PAT-NO: 5978804
DOCUMENT-IDENTIFIER: US 5978804 A

TITLE: Natural products information system

DATE-ISSUED: November 2, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Dietzman; Gregg R.	Friday Harbor	WA	98250	

APPL-NO: 8/ 833915
DATE FILED: April 10, 1997

PARENT-CASE:
This application claims benefit of provisional Application Ser. No. 60,015,286 filed Apr. 11, 1996.

INT-CL: [6] G06F 17/30
US-CL-ISSUED: 707/10; 707/104
US-CL-CURRENT: 707/10; 707/104.1; 707/506.1
FIELD-OF-SEARCH: 707/10, 707/100, 707/104, 707/103

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5179652	January 1993	Rozmanith et al.	345/331
☐	5241671	August 1993	Reed et al.	707/104
☐	5553277	September 1996	Hirano et al.	707/104
☐	5745895	April 1998	Bingham et al.	707/10
☐	5781773	July 1998	Vanderpool et al.	707/100

OTHER PUBLICATIONS

Duncan, "Publishing Databases on the World-Wide Web", PC Magazine pp. 403, 406-408, 410, and 412, Aug. 1995.

ART-UNIT: 271
PRIMARY-EXAMINER: Choules; Jack M.
ATTY-AGENT-FIRM: Seed and Berry LLP

ABSTRACT:

Disclosed is a data processing system for processing natural product information entered into the system using a standardized entry protocol. The data processing system stores data such as chemical structures, geographic

data processing system stores data such as chemical structures, geographic locations, taxonomy, genus synonyms, and textual descriptions and related natural products images such as images of the organisms, and geographic maps. The natural product images are correlated with the natural products data to allow display of the images with the related data. The data processing system further correlates the data products data and images stored in the system with remote databases, such as those containing existing commercially available data, linking the remote data thus correlated for display.

12 Claims, 30 Drawing figures

WEST

Generate Collection

L12: Entry 3 of 6 File: USPT Nov 2, 1999

DOCUMENT-IDENTIFIER: US 5978804 A
TITLE: Natural products information system

DEPR:
Software Versions for Windows 3.1 can be upgraded to NT. Programming for seamless application interface may become obsolete and impede functional development. For example, programming Paradox to emulate Object Link and Embed (OLE) to the image editor Adobe Photoshop.

DEPR:
Software--The software packages were assembled on both the Windows 3.1 and Windows NT 3.1 operating systems. Implementation of the total system requires exiting the applications to enter others, however, many features can be implemented using advanced Dynamic Data Exchange (DDE) technologies including Object Link and Embed (OLE). Query of the Sybase database tables is possible using Paradox initiated SQL Link to the Sybase Server. To determine an appropriate computer operating system for integration of the required software packages while considering the timing of software version release and rapid pace of software development required extra review and evaluation, the following software packages were used-Paradox for Windows V. 4.51 and Adobe Photoshop for Windows V. 2.5

DEPL:
Taxonomy--This screen-form table structure consists of seven tables that have a key index and secondary index on the NODC taxonomic code fields that link the tables in a hierarchy; three additional tables that contain information on synonyms and common names are linked by a key index to corresponding NODC taxonomic code and code suffixes. Taxonomic levels included are: Kingdom. Phylum, Class subclass, Order suborder, Family, genus species variant authority.

DEPX:
Taxonomy. Users are presented with menu driven hierarchical "entry pick" choices for data field entries. The tables that populate the menu choices are from the PSDE "standard checklist." For example, when the Kingdom Animalia is selected only the animal phyla are presented to the user for choices; when the Phylum Porifera is selected only the sponge classes are presented to the user for choices; this steps down to the species level where the species-variant-authority choices for a specific Genus are presented as choices.

DETL:
_____ System Requirements Hardware Environment Processor 486 or greater Memory 32 MB minimum, 64 MB recommended Disk Space 275 MB required (not including OS and supporting software), 1 GB recommended (for GIS) Virtual Memory (95) Windows Manage--50 MB minimum Page File (NT) 100 MB Release Media CD-ROM Software Environment Windows 95 Windows NT 4.0 (optional) Supporting Software Requirements Desktop Database Paradox v 7.0 (requires additional 25 MB disk) Customer Report Generator Crystal Reports v (optional--requires additional 8 MB disk) Chemical Compound Data and Structure Handling ISIS/Base and ISIS/Draw v 2.1 (optional--requires additonal 25 MB disk) Desktop GIS ArcView v 3.0, with Spatial Analysis (optional--requires additional 65 MB disk) Personnel Requirements Beta Test Site Manager responsible for communication with WPBM on bug reports and feature enhancement requests Beta Test Participants responsible for using NAPIS, testing the workflow in assigned modules, and creating bug reports for communication to WPBM Database Administration (optional) responsible for

communication to [redacted]M Database Administration (opt[redacted]al) responsible for migration of legacy datasets into NAPIS tables

WEST

Generate Collection

L12: Entry 4 of 6 File: USPT Oct 6, 1998

US-PAT-NO: 5818447
DOCUMENT-IDENTIFIER: US 5818447 A

TITLE: System and method for in-place editing of an electronic mail message using a separate program

DATE-ISSUED: October 6, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Wolf; Richard J.	Seattle	WA		
Koppolu; Srinivasa R.	Redmond	WA		
Raman; Suryanarayanan	Redmond	WA		
Rayson; Steven J.	Seattle	WA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Microsoft Corporation	Redmond	WA			02

APPL-NO: 8/ 660019
DATE FILED: June 6, 1996

INT-CL: [6] G06T 1/00
US-CL-ISSUED: 345/335; 395/683, 395/200.36, 707/516, 707/524
US-CL-CURRENT: 345/752; 345/853, 707/516, 707/524, 709/206
FIELD-OF-SEARCH: 395/682, 395/683, 395/200.33, 395/285, 395/200.6, 395/200.61, 345/335, 345/352, 707/516, 707/524, 707/530

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected | Search ALL

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5682532	October 1997	Remington et al.	395/683
☐	5682536	October 1997	Atkinson et al.	395/683 X
☐	5706458	January 1998	Koppolu	N/A

OTHER PUBLICATIONS

Oski, "Mac clients, take note: Lotus Notes 4.1 finally arrives", MacWEEK, v10, n21, pp. 23(2), May 1996.
Goulde, "Microsoft adapts to the WWW", Distributed Computing Monitor, v11 n2, pp. 21(5), Feb. 1996.
Jones, "Microsoft readies DocObject; technology will allow document editing in Web browsers", InfoWorld, v18 n18, p. 49, Apr. 1996.
Varhol, "Microsoft Activex will complement, not conquer, Java", Government Computer News, v15 n10, pp. 27 (2), May 1996.

Drews, "MIME interoperability: sharing files via E-mail", Network Computing, v7 n6, pp. 96 (2), Apr. 1996.
"OLE 2 Programmer's Reference", Microsoft Press, 1994, vol. 1, pp. 3-29.
"User's Guide: Lotus cc:MAIL, Release 2", Lotus Development Corporation, 1993, pp. 217-236.
"User's Guide: Lotus Ami Pro, Release 3.0", Lotus Development Corporation, 1993, pp. 429-441.

ART-UNIT: 272
PRIMARY-EXAMINER: Feild; Joseph H.
ATTY-AGENT-FIRM: Jones & Askew

ABSTRACT:

An email client invokes a DocObject-enabled mail note to display an email message and related features of the user interface. The mail note, which is a DocObject container, creates a DocObject server by invoking a DocObject-enabled word processor. The mail note provides a view port in which the word processor displays and edits the body of the email message. The word processor provides its formatting and editing features in the context of the mail note. OLE menu merging provides both email and word processing interoperability while editing the message. Programming interfaces between the mail note and the word processor allow the mail note to translate message data back and forth between the word processor's format and the format imposed by the email client. This ensures that messages created with the word processor can be read by other email clients.

40 Claims, 15 Drawing figures

WEST

Generate Collection

L12: Entry 4 of 6 File: USPT Oct 6, 1998

DOCUMENT-IDENTIFIER: US 5818447 A
TITLE: System and method for in-place editing of an electronic mail message using a separate program

BSPR:
Finally, a third approach has been one in which users have decided to use a full power word processor for authoring sophisticated and complex documents, and then use email for distribution. This requires the user to work in the word processing context to create and edit the document. When the document is complete, the user must switch to the email program, create a new message, and include the word processor document as an attachment. Although email is an effective mechanism for transporting documents, handling attachments requires several additional steps on the part of both the sender and the recipient of the message.

DRPR:
FIG. 3 illustrates a compound document in a word processing document frame.

DEPR:
OLE is a technology that enables developers to create extensible application programs that operate across multiple platforms. OLE-enabled applications allow users to manipulate information in an intuitive manner, using an environment that is more "document-centric" and less "application-centric." Users can create compound documents with data, or objects, of different formats, and focus directly on the data rather than on the application programs responsible for the data. The data can be embedded within the document, or linked to it, so that only a reference to the data is stored in the document.

DEPR:
The set of OLE services can be viewed as a two tier hierarchy. The lower level contains infrastructure services. These are basic services that provide the means by which features can be implemented and used. The infrastructure services include interface negotiation, memory management, error and status reporting, interprocess communication, structured storage, and data transfer. The upper level of the OLE service hierarchy provides application features, which are the services that benefit the end user. These include compound document management, in-place activation, programmability, and drag and drop operations.

DEPR:
All interface names are prefixed with either "I" or "IOle." Interfaces that begin with "IOle" provide services relating to compound document management. Those that begin with "I" provide services that are more general in nature. For example, IOleObject contains methods used by a client of an embedded or linked compound document object. IOleObject is implemented and used only by applications that participate in compound document management. IDataObject, however, contains methods that are used by all applications. These methods provide the means by which data of any type is transferred.

DEPR:
OLE supports the provision of a "compound document," which is a container object that contains a "linked" object or an "embedded" object. The difference between linked and embedded objects has to do with where the actual source data associated with the object is stored. This affects the object's portability, it method of activation and the size of the compound document.

DEPR:
When an object is linked, the source data continues to reside wherever it was initially created, which may be at another point in the document or in another, document altogether. Only a reference, or link, to the object is kept within the compound document. Linking is efficient and minimizes the size of the compound document. Changes made to the source are automatically reflected in any compound document that has a link to the source object. From the user's point of view, a linked object appears to be wholly contained within the document.

DEPR:
With an embedded object, a copy of the original object is physically stored in the compound document, along with all of the information needed to manage that object. As a result, the object becomes a physical part of the document. A compound document containing an embedded object will be larger than one containing the same objects as links. However, embedding offers advantages that offset the larger storage requirement. For example, compound objects with embedded objects can be transferred to another computer and edited there.

DEPR:
Embedded objects can be edited, or activated in place. This means that all maintenance of the object can be done without leaving the compound document. In order to edit the embedded object, the object must be explicitly activated or opened by performing an action such as double-clicking on the object's icon. This results in the object being displayed in a separate window with the user interface provided by the application program that created the object. The object is said to become in-place active (i.e., it is editable), and UI active (i.e., it displays the user interface associated with the application program that created the embedded object).

DEPR:
In summary, OLE allows objects to be embedded in a compound document, which is displayed in a container or frame. Generally, the embedded document is displayed in the container in what is referred to as object view. The container controls the appearance of the page and the layout of headers, footers, end notes, etc. The embedded object has no control over these aspects of the page. The container also controls the amount of space that is allocated to the embedded object for displaying its pictorial representation.

DEPR:
Some of the limitations associated with displaying an embedded object in a compound document are addressed by Microsoft Corporation's Document Object, or DocObject, technology. DocObject is an OLE 2.0 interface built on top of OLE and facilitates displaying an object in a "document view" instead of an "object view." The DocObject interface logically partitions a "view" of a document object from a "frame" in which the document object is displayed. The frame specifies the location and dimensions of a view port to which the document object is to display a view. The document view controls the page model and the dimensions of what is displayed within the view port.

DEPR:
Although using a full powered word processor in the context of a container mail note provides improved editing and formatting capabilities, those skilled in the art will understand that messages created in the word processor's native format must be compatible with a variety of email clients. The message format is defined by the email client. In an exemplary embodiment, the email client complies with the MAPI specification. The MAPI format ensures the interoperability between an embodiment of the present invention, a prior art rich text mail client, and other mail clients and gateways. This is particularly important with respect to an attachment. For example, a word processor may create a compound document in which the attached files are embedded in the document. This type of attachment handling is incompatible with MAPI, which requires that attachments be stored separately at the end of the message.

WEST

Generate Collection

L12: Entry 5 of 6 File: USPT Jun 2, 1998

US-PAT-NO: 5761684
DOCUMENT-IDENTIFIER: US 5761684 A

TITLE: Method and reusable object for scheduling script execution in a compound document

DATE-ISSUED: June 2, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Gibson; Kevin Patrick	Rochester	MN		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
International Business Machines Corporation	Armonk	NY			02

APPL-NO: 8/ 452791
DATE FILED: May 30, 1995

INT-CL: [6] G06F 9/40
US-CL-ISSUED: 707/515; 395/683
US-CL-CURRENT: 707/515; 709/318, 709/320
FIELD-OF-SEARCH: 395/700, 395/650, 395/600, 395/670, 395/672-675, 395/680, 395/683, 707/515

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4937743	June 1990	Rassman et al.	364/401
☐	5063523	November 1991	Vrenjak	364/514
☐	5148154	September 1992	MacKay et al.	340/712
☐	5339392	August 1994	Risberg et al.	395/161
☐	5537526	July 1996	Anderson et al.	707/515

OTHER PUBLICATIONS

Smith, Paul, "The OpenDoc Experience", EXE: The Software Developers Magazine, pp(6), Jul. 1994.
Smith, Paul, "SOM and OpenDoc", EXE: The Software Developers Magazine, pp(14), Feb. 1995.
Tycast, Robert L., "Component Software for the Masses: OpenDoc is Here", OS/2 Magazine, pp(3), Jul. 1994.
Adler, Richard M., "Emerging Standards for Component Software", IEEE Computer, pp(15), Mar. 1995.

Reeves et al., "A Distributed Object Road Map for Developers", OS/2 Magazine, pp(8), Sep. 1994.
"The Impact of Object-Orientation on Application Development" Cozkkuen A.A.R. IBM System Journal Sep. 1993 V32 N3 P420 (25).
R. Orfali et al., "Client/Server with Distributed Objects," BYTE, Apr. 1995, pp. 151-162.
"OpenDoc Shaping Tomorrow's Software," OpenDoc, 1993 Apple Computer, Inc., pp. 1-15.
R. Orfali et al., "Intergalactic Client/Server Computing," BYTE, Apr. 1995, pp. 108-122.
J. Mackenzie, "Document Repositories," BYTE, Apr. 1995, pp. 131-138.
E. X. Dejesus, "Dimensions of Data," BYTE, Apr. 1995, pp. 139-148.

ART-UNIT: 275
PRIMARY-EXAMINER: Oberley; Alvin E.
ASSISTANT-EXAMINER: Courtenay, III; St. John
ATTY-AGENT-FIRM: Meador; Terrance A.

ABSTRACT:

A reusable script execution scheduling part for compound documents in a document-centric processing environment. Document-centric computing environments having architectures similar to OpenDoc.TM. include a technique for executing scripts to interact with the compound document content. The CHRON part of this invention includes embedded objects for defining scheduled execution times for one or more scripts that may be opened within the compound document to provide a view of its contents. Either the ScheduleTime element or the ScriptEvent element of the CHRON part may be opened and edited in place. The reusable CHRON part sets up event scheduling with the operating system so that specified scripts are run according to specified ScheduleTimes.

18 Claims, 21 Drawing figures

Generate Collection

L12: Entry 5 of 6 File: USPT Jun 2, 1998

DOCUMENT-IDENTIFIER: US 5761684 A
TITLE: Method and reusable object for scheduling script execution in a compound document

ABPL:
A reusable script execution scheduling part for compound documents in a document-centric processing environment. Document-centric computing environments having architectures similar to OpenDoc.TM. include a technique for executing scripts to interact with the compound document content. The CHRON part of this invention includes embedded objects for defining scheduled execution times for one or more scripts that may be opened within the compound document to provide a view of its contents. Either the ScheduleTime element or the ScriptEvent element of the CHRON part may be opened and edited in place. The reusable CHRON part sets up event scheduling with the operating system so that specified scripts are run according to specified ScheduleTimes.

BSPR:
This invention relates generally to event scheduling in object-oriented application systems and specifically to a reusable compound document script scheduling part that provides class methods for the scheduled execution of embedded scripts in an object-oriented compound document processing system.

BSPR:
With the development of the CORBA specification and the related Interface Definition Language (IDL) specification, several new object-oriented system environments have been developed by practitioners in the art, including a class of environments herein denominated "document-centric" processing system. The emergence of document-centric computing technology represents a major event in the transition to object-oriented technology from process-oriented technology. The document-centric computing art now includes at least three commercially-available compound document architectures. These include Microsoft's Object Linking and Embedding (OLE.TM.) System, the Taligent.TM. consortium's compound document framework, and the Component Integration Laboratory (CI Labs) OpenDoc.TM. compound document application architecture. CI Labs is an independent industry consortion responsible for the OpenDoc architecture. Reference is made to the OpenDoc white paper (OpenDoc: Shaping Tomorrows Software, Apple Computer, Inc., Cupertino, Calif., 1993) for an introduction to the OpenDoc architecture.

BSPR:
Major elements of the OpenDoc architecture include objects for "documents," "parts," "part handlers," and "frames." The OpenDoc "document" is a "compound document" and is fundamentally different from the usual meaning of the term document. A compound document is no longer a single block of content bound to a single application but is instead composed of smaller blocks of content herein denominated "parts." Parts are the fundamental building blocks of the OpenDoc architecture, replacing monolithic applications with smaller units of content dynamically bound to related functionality. OpenDoc parts each contain data. For example, text parts contain characters, graphics parts contain lines and shapes, spreadsheet parts contain spreadsheet cells with formulas, and video parts contain digitized video. The particular data type within each part is defined by the part developer and is denominated the part's "intrinsic content." In addition to intrinsic content, a part may contain other parts, each having its own intrinsic content. Every compound document has a single "root part" at the top level into which all other parts are embedded. Usually if a part can contain one type of part, it may contain all conforming types of

parts, including OpenDoc parts yet to be developed.

BSPR:
Parts may also be viewed as the boundaries at which one kind of content in a document ends and another begins. Each part of a document has its own content model, which is the model of objects and operations that is presented to the user. The content model changes at the "frame" between parts in a compound document. Frames are areas of the display that represent a part but also serve as a handle permitting manipulation of the part as a whole as well as permitting the user to see and edit the intrinsic contents of a part. A frame is much more than a standard application window, which is only visible when the part is being viewed or edited. A frame is persistent and remains visible when opened into a window. When the window is closed, the part returns to the representation from which it was opened and remains as a persistent element of the compound document.

BSPR:
In the OpenDoc architecture, "part handlers" are the rough equivalent of application programs. Part handlers include part editors and part viewers and may be responsible for displaying the part both on the screen and for printing purposes, editing the part, and storage management (both persistent and runtime) for the part. The part handler must read the part from persistent storage into main memory, manage the runtime storage associated with the part and write the part back out to persistent storage when appropriate. Part handlers are dynamically linked into the runtime world of the compound document depending on the part types that appear in the compound document.

BSPR:
The OpenDoc architecture storage model is based on the Apple Computer, Inc. Bento.TM. Object Container System (OCS) that functions as the reference file storage system, the object container for parts placed on a clipboard, and the container for transporting documents across platforms. The OpenDoc architecture also provides the Open Scripting Architecture (OSA) for scripting languages such as OREXX, AppleScript, ScriptX, and OLE Automation. OSA includes script typing that permits identification of the correct scripting engine for script processing and standardization of script semantic messages according to the Open Events form required by OSA. Scripts may become a part of the compound document, either attached or embedded as necessary. They may also operate as control structures, as front- or back-ends to operations or as instructions for use embedded into a compound document for transfer across system boundaries. Parts that are OSA-scriptable must register their capabilities by declaring the event suites that they support. Scripts are then built for a particular set of event suites and those scripts then may run on all parts that support those particular suites.

BSPR:
Another problem felt in the compound document art is the lack of any useful method for scheduling a point in time for executing an OSA script in a compound document. This particular problem results in part from the power of the OSA, with which scripts may be either attached or embedded in compound documents or in other parts as desired and thereafter carried along across boundaries throughout a distributed system with the parent part or document. Scripts may be executed either in response to a user command or in response to a message from another object, but execution of a script at a predetermined date and time is more problematic. The event scheduling element of the operating system must first create and store a persistent log entry specifying the date and time for execution of the script in the system event log. If a special-purpose application program is used to create the necessary log entry, the resulting application object does not provide for "reuse," thereby increasing programmer workload and system complexity. If the necessary date and time commands are included in the script itself, then the script cannot be "reused" and must be revised for each desired execution data and time. That is, the usual process-centric solutions to script-execution scheduling are not effective in the document-centric system model, and no user alternative techniques have been suggested by practitioners in the art until now.

BSPR:
In U.S. Pat. No. 4,937,743, Rassman et al. disclose a project resource scheduling GUI system that provides automatic conflict resolution, periodic

monitoring and alarm-triggering of scripts. Again, although Rassman et al. consider the dynamic management of a plurality of resources, they neither consider nor suggest reusable objects for object-oriented processing in a document-centric system architecture. In U.S. Pat. No. 5,063,523, Vrenjack discloses a data communication network management system that permits a user to establish rules for pattern-matching to selected attributes of incoming events, such as alarms, from network objects. Although Vrenjack describes a process-centric system including the well-known concept of an event that can trigger a script, where the event may also be date or time based, he neither considers nor suggests any method for providing a reusable compound document script scheduling part suitable for document-centric applications.

BSPR:
Accordingly, there is still an unmet need for a class type or method to run scripts at predetermined times in a document-centric computing environment such as the OpenDoc architecture. Such a method must operate within the context and limitations of the compound document without unnecessarily avoiding the advantageous features of the document-centric computing environment. The related unresolved problems and deficiencies are clearly felt in the art and are solved by this invention in the manner described below.

BSPR:
This invention solved the script execution scheduling problem by introducing a new class type, herein denominated the CHRON type, for scheduling script execution in a compound document. Because each "instance" of the CHRON class is an object, it provides the reusability and extensibility features necessary for proper integration into a document-centric architecture such as the OpenDoc architecture. The CHRON part defines the point in time at which a script is to be executed and includes one or more embedded script parts that specify the scripts to be executed, one or more embedded times/date parts that specify when the scripts are to be executed and may also contain embedded tabular specifications of relationships between time parts and script parts. The CHRON part is represented to the user when minimized by an icon that displays the predetermined script execution time.

BSPR:
It is an object of this invention to provide a reusable compound document script scheduling part for automatic script execution at predetermined times. It is a feature of the CHRON part of this invention that it is both reusable and extensible in accordance with object-oriented programming architectural requirements. It is an advantage of the CHRON part of this invention that it may be manipulated by the user through a Graphical User Interface (GUI) to quickly and easily revise either the designated scripts or the time/date parts that specify when the scripts are executed.

DRPR:
FIG. 5 shows an example of a compound document in accordance with the OpenDoc.TM. Architecture from the prior art;

DRPR:
FIG. 7 is a functional block diagram of an illustrative embodiment of a compound document including the CHRON part of this invention;

DEPR:
The class type and object of this invention may be better understood in view of a brief discussion of the object-oriented programming art in general and compound document architecture in particular. With the advent of object-oriented programming techniques leading to compound document architectures such as OpenDoc and OLE and the document framework in Taligent, the computing model for the user has evolved from a "tool-centric" to a "document-centric" model. In the document-centric model of computing, the user may focus on the document desired rather than on the output of some specific tool or application program. To the user, a document is now composed of several "parts" that are each embedded and integrated into a "document." The parts are "class types" or objects that may now be completely incorporated into the document. Part usefulness depends only on availability of the necessary part handler application (e.g., part editors and part viewers).

DEPR:

Compound Document Architecture In General

DEPR:
The generic document-centric system providing for the embedding of objects into a document is embodied in an object-oriented system such as the OpenDoc or Taligent document framework. The fundamental concept of the compound document is that it can hold different kinds of data in the form of individual document parts and that each kind of part is handled by an independent application or part handler. Each part handler independently understands its own intrinsic content and need not know anything about the intrinsic content of other parts embedded therein. The integration and cooperation of these parts is accomplished by the compound document system architecture, policies and protocols.

DEPR:
FIG. 1 shows the class library of objects (SOM or CORBA objects) employed by the OpenDoc system to realize the basic interfaces necessary in a compound document architecture. FIG. 1 also shows the inheritance relationships of the OpenDoc class library known in the art. FIG. 2 shows the runtime relationships of the primary OpenDoc object type, which is more meaningful to the system developer. FIGS. 1 and 2 exemplify one embodiment (the OpenDoc system) of the general compound document architecture, which is now discussed.

DEPR:
A compound document consists of parts embedded in a document container. The content of any particular part is not limited because each part is isolated from the other by the document architecture. The only practical limitation on content is the availability of a part handler, either editor or viewer. As new part handlers are created, they may be immediately used to manage the appropriate parts. The user may rely on a clipboard and on drag-and-drop mechanisms within the document. Part handlers may be changed or replaced to fit specific user requirements, which may be satisfied independently of existing documents because these do not depend on any specific part handler. The different parts are isolated by using linkages for data transfer and navigation, which may be both internal to the compound document (such as hypertext-type links) and external to the compound document (for data transfer and navigation).

DEPR:
The fundamental problems faced in a compound document system include data exchange, part registration, part-to-part communication, document layout, user interface control, events and messages, document versioning and document storage. Many compound documents are built-up by the movement of data from one document (or the desktop) to another either through a cut-and-paste or a drag-and-drop operation by the user. These interfaces determine how data is moved from one document to another and must therefore define how data is added and removed from the clipboard. The interfaces must also transfer type and attribute information so that the proper part handler application program can be retrieved from the part handler database and launched. Because the clipboard must transfer complex data-objects, an in-memory object container is also required. The storage format standard for clipboard data in OpenDoc is the Bento container shown in FIG. 3.

DEPR:
With a number of embedded parts sharing space on the screen and in a printed document, central coordination is required over display layout. The display layout controls operate with document window geometry and the geometry of embedded parts, which contains information about the size, shape, position, clip extents and transformations of the frames and windows together with the means for establishing and manipulating them. A document layout interface must handle the negotiation for space resources between the part handler and the container. It is used when the part must grow or shrink in size in response to user interactions such as picture editing or data changes arising from a link update of embedded data. The part container establishes the policy for document layout and has final authority therefor. Such negotiations may result in the part being placed on a new page, or being split into more than one piece or in the rearrangement of the document to accommodate a new part size or even in refusal of authority for the part to change size.

DEPR:
Document collaboration and versioning interfaces manage multiple drafts or versions of a compound document and must provide for the election and creation of a compound document draft together with management and reconciliation of multiple drafts. These are closely related to document storage interfaces, which manage the storage and retrieval of a compound document and its parts. Document storage interfaces must be abstracted from the underlying storage systems sufficiently to support a multiplicity of different storage systems. In the OpenDoc reference implementation, the storage system is constructed using Bento containers, for instance.

DEPR:
OpenDoc.TM. Compound Document Part Interfaces

DEPR:
FIG. 3 shows the Bento object container 56 known in the art. Bento container 56 can be thought of as an "envelope" for objects and it may be generally applied to any object-oriented programming system, including the OpenDoc document-centric programming system. Bento container 56 includes standard procedures for moving itself, for viewing the inventory list, for removing things from and adding new things into the envelope. All such procedures are independent of container contents. In a document-centric architecture, Bento container 56 serves three functions. It is the reference file storage system, the object container for things placed on the clipboard and the container of choice for transporting compound documents across platforms.

DEPR:
FIG. 4 shows the structure of the OpenDoc StorageUnit class type embodied as object 50. Object 50 includes two "Property" objects 70 and 72. Property object 72 includes two embedded Value objects 74 and 76. Value object 76 is linked to another StorageUnit object 78, which is itself embedded in the same current draft object 80 containing StorageUnit 50. StorageUnit object 78 is also linked to two other commonly-embedded StorageUnits 82 and 84.

DEPR:
FIG. 5 shows a "canonical" compound document display 86 according to the OpenDoc model, which enables the creation of compound documents that are created and edited by several cooperating applications working within a single compound document.

DEPR:
The key to the notion of parts is that each part of a compound document has its own "content model," which is the model of objects and operations that is presented to a user of the part. The content model changes at the boundary between parts. For instance, in document 86, text part 98 includes lines, words, paragraphs, characters, and embedded button part 100. Internally, the part handler in charge of part 98 may have a model that includes run-length encoded arrays for style information, line end arrays and similar tools. These are not presented to the user so they are not part of the content model. In the root graphics part of the compound document, the content objects may look very different. Circles, rectangles, and lines are content objects that could be provided in such a graphics part.

DEPR:
A part and its handler together form the equivalent of a programmatic object (data and methods) in the object-oriented programming sense. The part provides the state information while the part-handler provides the methods or behavior. When bound together, they form an editable segment of a compound document. Part handlers are dynamically linked into the runtime world of the compound document based on the part types that appear in the document. Because any type of part may appear in any document at any time, the part handlers must be dynamically linked to provide a smooth user interface. The OpenDoc system assumes that parts are mainly used in a shell that allows a compound document to act like a single application. This document shell provides an address base, distributes events and provides basic user interface resources such as windows and menus.

DEPR:
Because one of the important features of the OpenDoc system is customized

documents, the part handlers must handle two distinct kinds of events: semantic events and user-interface (UI) events. The difference between the two event types relates to whether the event makes sense outside of the particular document display context. A UI event requires information about where windows are located in the display, how parts of the document have been scrolled, or in what part a selection was last made. These events include mouse clicks, keystrokes and menu activations. In contrast, a semantic event relates to the semantics of the compound document and is generally independent of the graphical context. Semantic events relate to the content model of the part.

DEPR:
Given the presence of multi-part compound documents, a persistent storage mechanism is needed to enable multiple part handlers to share a single compound document file. OpenDoc assumes that such a storage system can effectively give each part its own storage stream and that reliable references can be made from one such stream to another. Because many code sections may need access to a given part, the storage system must support a robust annotation scheme allowing information to be associated with a part without disturbing the part format.

DEPR:
At runtime, OpenDoc assumes that an instance of the document shell is created for each document. This generic shell must provide four basic structures to the part handlers: (a) the storage system, (b) the window and its associated state, (c) the event dispatcher and (d) an arbitration registry for negotiating shared resources such as the menu bar. The runtime shell may also be responsible for binding and loading part handlers for the parts that appear in the compound document. It is assumed that once a given part handler is loaded, any part in any document may share the executable code of the part handler.

DEPR:
Because OpenDoc separates the part handlers from document level functions, new features can be added to compound documents without revising of existing applications. OpenDoc's document shell thus provides access to existing applications without modification. OpenDoc also allows for collaboration between parts through scripting. Scripting forms a rich medium for coordinating the work of parts in documents and allows users and parts to work together to perform tasks.

DEPR:
The Open Scripting Architecture (OSA) provides an open architecture for scripting languages such as OREXX, Apple Script, OLE Automation, and Scriptx. The OSA includes two components. The first is the typing of script types that permits inclusion of the correct scripting engine for script processing. The second is the standardization of script semantic messages in the Open Event form. Scripts become part of a compound document, either attached or embedded as necessary. They may be used to implement control structures or used as instructions for embedding into documents that are mailed or transferred out of the network.

DEPR:
The Open Events model is based on the standard registry of verbs and object classes. Open Events are arranged in application suites that include such application groups as "Required," "Core," "Text," "Table," "Database," "Compound Document," and many others yet to be invented. Each event includes three elements. These are (a) event verbs such as "Open," "Close," "Select," "Get," "Put," and the like, (b) object classes or object specifiers such as "Application," "Document," "Word," "Paragraph," "Circle," and the like, and (c) descriptor types such as Boolean Fixed Attribute "greater than," "3.sup.rd," "bold," and the like.

DEPR:
A new compound document part, herein denominated the CHRON part, allows the user working on a compound document to specify "script events" with which the operating system automatically executes designated scripts at predetermined times. FIG. 7 shows a compound document object 120 that includes two CHRON objects 122 and 124. Document 120 also by way of example is shown having an embedded text part 126, an embedded video part 128, a table part 130, and an

image part 132. Each of the parts within document 120 may be opened by the user to view or edit its contents. The document-centric system automatically links each object to the part handler necessary for editing and/or viewing the content of the object. For instance, text object 126 can be opened by the user with a mouse click. The system then may respond to the mouse click by launching a part handler to display the text content of object 126 and/or launching a text editing application program to accept keyboard and mouse editing commands from the user. Both display and editing can be limited to a predefined window in the user display.

DEPR:
The CHRON class type also includes one or more scheduled-time parts (time/date), as exemplified by scheduled-time parts 134 and 136, each of which specifies when a script is to be executed. CHRON part 122 may also include a table part (not shown) that specifies the relationship between the two scheduled-time parts 134 and 136 and the single script part 138, for example. An instance of the CHRON class type may be inserted into any container part within a compound document, including the root document itself exemplified by document 120. Thus, although not shown, if script event 142 in CHRON part 124 causes the execution of a word search in the content model of text part 126, then CHRON part 124 could be embedded within text part 126 itself rather than in root document 120.

DEPR:
While the invention discussed above is primarily disclosed as a process, practitioners of ordinary skill in the art can appreciate that an apparatus or system, such as the system illustrated in FIG. 11, can be programmed or otherwise designed to facilitate the practice of the process of this invention. For instance, system 200 in FIG. 11 may include a central processing Unit (CPU) 202 linked to a graphical user interface (GUI) display 204 and a memory system 206. The system user may view and edit compound documents from a keyboard 208 and a mouse 210 in combination with GUI display 204. Memory 206 may be linked to a persistent data storage subsystem 212 in the usual manner. In a compound document system, memory 206 may include numerous program objects, exemplified by a storage manager program object 214 for controlling data transfers between memory 206 and persistent memory 212, an operating system object 216 for controlling all system activities, and a plurality of class type instances, otherwise herein denominated "objects," exemplified by object 218. By way of example, object 218 may represent a compound document including an instance of the CHRON class type of this invention. It can be understood and appreciated that a computer system exemplified by system 200 in FIG. 11 also falls within the spirit and scope of this invention.

CLPR:
1. In a machine-implemented document-centric application processing system including a data processor coupled to a user display, to means for accepting user requests and to memory means including persistent storage means for storing an object-oriented operating system and a class library of objects including a plurality of data objects and a plurality of program objects, said memory means including a script-editing program object, a schedule-time object editor, an icon-display program object, an event-scheduling system and a plurality of parts and part editors, a machine-executed method for scheduling script execution in a compound document object, said method comprising the steps of:

CLPR:
5. A reusable compound document script scheduling object for scheduling script execution in a compound document object in a machine-implemented document-centric application processing system including a data processor coupled to a user display, to means for accepting user requests and to memory means including persistent storage means for storing an object-oriented operating system and a class library of objects including a plurality of data objects and a plurality of program objects, said memory means including a script-editing program object, a schedule-time object editor, an icon-display program object, an event-scheduling system and a plurality of parts and part editors, said reusable script scheduling object comprising:

CLPR:

10. An object-oriented compound document processing system comprising:

CLPR:
11. The object-oriented compound document processing system of claim 10 further comprising:

CLPR:
12. The object-oriented compound document processing system of claim 11 wherein each said CHRON icon schedule field comprises:

CLPR:
13. The object-oriented compound document processing system of claim 12 wherein said GUJI linkage means includes means for launching said script-editing program object to edit said script object responsive to a user request.

CLPR:
14. The object-oriented compound document processing system of claim 11 wherein said GUI linkage means includes means for launching said script-editing program object to edit said script object responsive to a user request.

CLPR:
15. A computer program product for use in scheduling script execution in compound document objects in a machine-implemented document-centric application processing system including a data processor coupled to a user display, to means for accepting user requests and to memory means including persistent storage means for storing an object-oriented operating system and a class library of objects including a plurality of data objects and a plurality of program objects, said memory means including a script-editing program object, a schedule-time object editor, an icon-display program object, an event-scheduling system and a plurality of parts and part editors, said computer program product comprising:

CLPV:
(a) spawning in said memory means an instance of a reusable CHRON object including at least one script object and at least one schedule-time object, said instance being contained in said compound document object;

CLPV:
object container means stored in said memory means for containing said at least one script object and said at least one schedule-time object and for linking the components of said reusable script scheduling object with other said parts contained in said compound document; and

CLPV:
a plurality of compound documents each containing one or more parts in said memory means;

CLPV:
one or more reusable script-scheduling objects each for scheduling script execution in a compound document;

CLPV:
means recorded on said recording medium for directing said document-centric application processing system to spawn in said memory means an instance of a reusable CHRON object including at least one script object and at least one schedule-time object, said instance being contained in said compound document object;

WEST

End of Result Set

Generate Collection

L12: Entry 6 of 6 File: USPT Dec 16, 1997

US-PAT-NO: 5699518
DOCUMENT-IDENTIFIER: US 5699518 A

TITLE: System for selectively setting a server mode, evaluating to determine server node for executing server code, and downloading server code prior to executing if necessary

DATE-ISSUED: December 16, 1997

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Held; Andrew G.	Kirkland	WA		
Jung; Edward	Seattle	WA		
Zbikowski; Mark	Woodinville	WA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Microsoft Corporation	Redmond	WA			02

APPL-NO: 8/ 158631
DATE FILED: November 29, 1993

INT-CL: [6] G06F 13/00, G06F 15/76
US-CL-ISSUED: 395/200.11; 395/335, 395/653
US-CL-CURRENT: 709/229; 709/225
FIELD-OF-SEARCH: 395/200, 395/575, 395/650, 395/200.11, 395/335, 395/653

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected | Search ALL

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4887204	December 1989	Johnson et al.	364/200
☐	5157390	October 1992	Yoshie et al.	340/825.52
☐	5167035	November 1992	Mann et al.	395/575
☐	5287537	February 1994	Newmark et al.	395/800
☐	5325527	June 1994	Cwikowski et al.	395/650
☐	5329619	July 1994	Page et al.	395/200
☐	5341477	August 1994	Pitkin et al.	395/200
☐	5390297	February 1995	Barber et al.	395/200
☐	5430876	July 1995	Schreiber et al.	395/650
☐	5440744	August 1995	Jacobson et al.	395/650

OTHER PUBLICATIONS

"OS/2 Multiple Specification Within Path and File Name", IBM TDB n5 10-91 p223 Order 91A 62640, Oct. 1991.
"The Annotated C++ Reference Manual", Ellis et al, ANSI Base Document, 1990.
Microsoft LAN Version 2.0 User's Guide for MS OS/2, Microsoft Corporation 1990 pp. 45, 140-141.
"The Design and Implementation of a Reliable Distributed Operating System--ROSE", Ng, IEEE, pp. 2-11, Apr. 1990.
"A service Platform for Distributed Applications", Popescu-Zeletin et al, IEEE, pp. 11-17, Jul. 1992.
Microsoft Windows Version 3.1 Programmer's Reference vol. 1: Overview, Microsoft Corporation 1992, pp. 210-212, 232.
Microsoft Windows Version 3.1 Programmer's Reference vol. 2: Functions, Microsoft Corporation 1992, pp. 672-683, 685-686, 688-689, 695-698, 712-713, 722.
Microsoft LAN Manager Version 2.0 Administrator's Guide, Microsoft Corporation 1990, pp. 235-245.
Vinoski, Steve, "Distributed Object Computing with CORBA", C++ Report 5:32-38, Jul.-Aug. 1993.
"Efficient Message Dispatching in Distributed Environments", IBM Technical Disclosure Bulletin 35:437-438, 1992.
Mowbray, T.J. et al., "Interoperability and CORBA-based Open Systems", Object Magazine 3:1055-3614, Sep.-Oct. 1993.
Arnold, J. et al., "Control Integration and Its Role in Software Integration", Genie Logiciel & Systemes Experts 30:14-24, Mar. 1993.
Tanenbaum, A.S. et al., "The Amoeba Distributed Operating System--A Status Report", Computer Communications 14:324-335, 1991.
Varadharajan, V. et al., "Multilevel Security in a Distributed Object-Oriented System", Computers & Security 10:51-68, 1991.

ART-UNIT: 237
PRIMARY-EXAMINER: Lee; Thomas C.
ASSISTANT-EXAMINER: Perveen; Rehana
ATTY-AGENT-FIRM: Seed and Berry LLP

ABSTRACT:

A method and system for executing code remotely is provided. In a preferred embodiment, a client program executes on a client node and communicates with a network. The executing client program then requests the execution of server code corresponding to an object instance or object class instance with which the client program desires to communicate. In response to the client program request, the computer system determines a location where the server code will be executed. This determination is made using a set of rules that arbitrate between location contexts specified by the corresponding server program and a location context requested by the client program. Once the appropriate location is determined, the client program forwards its request to the appropriate

network node, which requests execution of the requested server code.

34 Claims, 17 Drawing figures

WEST

End of Result Set

Generate Collection

L12: Entry 6 of 6 File: USPT Dec 16, 1997

DOCUMENT-IDENTIFIER: US 5699518 A
TITLE: System for selectively setting a server node, evaluating to determine server node for executing server code, and downloading server code prior to executing if necessary

BSPR:
An advantage of using object-oriented techniques is that these techniques can be used to facilitate the sharing of data and code. In particular, object-oriented techniques facilitate the creation of compound documents. A compound document is a document that contains objects generated by various computer programs. (Typically, only the data members of the object and the class type are stored in a compound document.) For example, a word processing document that contains a spreadsheet object generated by a spreadsheet program is a compound document. A word processing program allows a user to embed a spreadsheet object (e.g., a cell) within a word processing document. To allow the embedding of a spreadsheet object, the word processing program needs to be compiled with the class definition of the spreadsheet object to enable the word processing program to invoke function members of the spreadsheet object. To allow objects of an arbitrary class to be embedded into compound documents, interfaces are defined through which an object can be accessed without the need for the word processing program to have access to the class definitions at compile time. An abstract class is a class in which there is at least one virtual function member with no implementation (a pure virtual function member). An interface is an abstract class with no data members and whose virtual functions are all pure. Thus, an interface provides a protocol for two programs to communicate. Interfaces are typically used for derivation: a program implements classes that provide implementations for the interfaces the classes are derived from. Thereafter, objects are created as instances of these derived classes.

DEPR:
In the case of a compound document system, the client program implements the compound document and the server program provides server code to manage embedded (or linked) objects stored within the compound document. Referring to the example described in the Background section, the word processing program is the client program, which provides support for the compound document. The embedded spreadsheet object is supported and manipulated by the spreadsheet program, which is the server program.

DEPR:
In addition to the location of server code, a server program can specify in the persistent registry that it wants the server code to be run in a particular security context. A security context is used by the computer system to ensure that only certain code can access data, code, and system resources that require a defined level of authorization before access is permitted. If the code to be executed has a security context with the proper authorization specified, then access to protected data, code, and resources is permitted. A security context specification is typically operating system dependent. For example, a security context can be viewed as a user account (user ID) and a password. There are many ways to implement the specification of a security context. In one embodiment, the server code specifies user IDs in a persistent registry and corresponding passwords in a secure database. When a network service or client program executes server code, it uses the information in the persistent registry and secure database to execute the server code in a particular security context.